Flexible Real-Time Reverberation Synthesis with Accurate Parameter Control

Prawda, Karolina; Willemsen, Silvin; Serafin, Stefania; Välimäki, Vesa

Published in:
Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

FLEXIBLE REAL-TIME REVERBERATION SYNTHESIS WITH ACCURATE PARAMETER CONTROL

Karolina Prawda *

Acoustics Lab
Dept. of Signal Processing and Acoustics
Aalto University, Espoo, Finland
karolina.prawda@aalto.fi

Silvin Willemsen, Stefania Serafin

Multisensory Experience Lab
Dept. of Architecture, Design & Media Tech.
Aalborg University, Copenhagen, Denmark
{sil,sts}@create.aau.dk

Vesa Välimäki

Acoustics Lab
Dept. of Signal Processing and Acoustics
Aalto University, Espoo, Finland
vesa.valimaki@aalto.fi

ABSTRACT

Reverberation is one of the most important effects used in audio production. Although nowadays numerous real-time implementations of artificial reverberation algorithms are available, many of them depend on a database of recorded or pre-synthesized room impulse responses, which are convolved with the input signal. Implementations that use an algorithmic approach are more flexible but do not let the users have full control over the produced sound, allowing only a few selected parameters to be altered. The real-time implementation of an artificial reverberation synthesizer presented in this study introduces an audio plugin based on a feedback delay network (FDN), which lets the user have full and detailed insight into the produced reverb. It allows for control of reverberation time in ten octave bands, simultaneously allowing adjusting the feedback matrix type and delay-line lengths. The proposed plugin explores various FDN setups, showing that the lowest useful order for high-quality sound is 16, and that in the case of a Householder matrix the implementation strongly affects the resulting reverberation. Experimenting with delay lengths and distribution demonstrates that choosing too wide or too narrow a length range is disadvantageous to the synthesized sound quality. The study also discusses CPU usage for different FDN orders and plugin states.

1. INTRODUCTION

Artificial reverberation is one of the most popular audio effects. It is used in music production, sound design, game audio, and movie production to enhance dry recordings with the impression of space. The development of digital artificial reverberation started nearly 60 years ago [1], and since then various improvements as well as different techniques have been developed [2]. The designs available nowadays can be roughly divided into three groups: convolution algorithms, delay networks, and physical room models [2, 3, 4].

The methods involving physical modeling simulate sound propagation in a specific geometry. Due to their high computational cost, though, they are used mostly in off-line computer simulations of room acoustics [3]. Recent developments in hardware and software technologies have also allowed computationally expensive simulations, such as those based on 3-D finite-difference schemes, to run in real time [5].

* This work was supported by the "Nordic Sound and Music Computing Network—NordicSMC", NordForsk project number 86892.

The techniques convolving the input signal with a measured room impulse response (RIR) produce rich, high-fidelity reverberation. However, since the RIR samples serve as the coefficients of a finite impulse-response (FIR) filter, with which the dry signal is filtered, the computational cost is high, especially for long RIRs.

Another group of artificial reverberation algorithms is based on networks of delay lines and digital filters. The first example of such reverberators was introduced by Schroeder and Logan [1], who used feedback-comb-filter structures to create a sequence of decaying echoes. A similar architecture using allpass filters was also proposed to ensure high echo density without spectral coloration. The development of such structures led to the invention of feedback delay network (FDN) algorithms, which can be regarded as a "vectorized" comb filter [2]. The FDN, as used in its current form, was presented in the work of Jot and Chaigne [6, 7].

Over the years, many real-time implementations of artificial reverberation algorithms have been developed. The designs that use a convolution-based approach, however, depend on measured or pre-synthesized RIRs convolved with the signal, which are collected in groups of presets [3, 8, 9, 10]. Such Virtual Studio Technology (VST) plugins allow modifying the reverberation by modulating, damping or equalizing the available RIRs. The possibilities are, however, limited by the size of the RIR databases and therefore prove to be relatively inflexible.

Algorithmic reverb plugins that are based on delay network designs are both computationally efficient and easily modulated, thus providing more flexibility and freedom in producing reverberated sounds [4, 11]. The available designs vary between simple solutions allowing the user to change only a few parameters [12] and complex architectures with an elaborate interface enabling control over a wide range of variables [13]. Many of those plugins, however, still remain ambiguous about the reverberation they synthesize, allowing the user to set only the broadband decay parameter, and rely on presets based on the types of rooms they are supposed to imitate (e.g., Bright Room or Dark Chamber [14]). Usually, they also lack the information about the reverberation algorithm they use and its elements.

The present work proposes a real-time implementation of an FDN algorithm with accurate control over the reverberation time (RT) in ten octave frequency bands in the form of an audio plugin. The graphical user interface (GUI) gives a thorough insight into the attenuation filter's magnitude response, corresponding RT curve, and resulting impulse response (IR). The plugin also provides several possibilities to control the elements of the FDN structure, such as the feedback matrix and delay lines. It gives the user a full view of the decay characteristics and quality of the synthesized reverberation. The study also presents the effect that the type and size of the feedback matrix and the lengths and distribution of

the delay lines have on the produced sound and the algorithm's performance.

This paper is organized as follows: Section 2 presents the theory behind the FDN, and Section 3 shows the GUI of the implemented plugin, describes the functionalities and user-controlled parameters of the reverberator, presents the code structure and discusses the real-time computation issues. Section 4 shows and discusses results regarding the echo density produced by the implementation and the CPU usage of the plugin. Finally, Section 5 summarizes and concludes the work.

2. FEEDBACK DELAY NETWORK

Figure 1 presents a flow diagram of a conventional FDN, which is expressed by the relation:

$$y(n) = \sum_{i}^{N} c_i s_i(n) + dx(n), \tag{1a}$$

$$s_i(n + L_i) = \sum_{j}^{N} A_{i,j} \widetilde{h}_i(n) s_j(n) + b_i x(n), \tag{1b}$$

where $y(n)$ and $x(n)$ are the output and input signal, respectively, at time sample n, $s_i(n)$ is the output of the ith delay line, and $A_{i,j}$ is the element of an N-by-N feedback matrix (or scattering matrix) $\mathbf{A}$, through which all the delay lines are interconnected. Parameters b_i and c_i symbolize input and output coefficients, respectively, d is the direct-path gain, and $\widetilde{h}_i(n)$ is the attenuation filter of the ith delay line.

When designing an FDN, a common practice is to first ensure that the energy of the system will not decay for any possible type of delay. Therefore, the matrix $\mathbf{A}$ should be unilossless [15]. To obtain a specific frequency-dependent RT, each of the delay lines must be cascaded with an attenuation filter, which approximates the target gain-per-sample expressed by

$$\gamma_{\mathrm{dB}}(\omega) = \frac{-60}{f_{\mathrm{s}} T_{60}(\omega)}, \tag{2}$$

where $T_{60}(\omega)$ is the target RT in seconds, $\omega = 2\pi f / f_{\mathrm{s}}$ is the normalized angular frequency, f is the frequency in Hz, and f_{s} is the sampling rate in Hz. In order to ensure that all delay lines approximate the same RT, the gain-per-sample for each of them must be scaled by a respective delay in samples L. This implies that the target magnitude response of the attenuation filter in dB is defined as follows:

$$A_{\mathrm{dB}}(\omega) = L \gamma_{\mathrm{dB}}(\omega). \tag{3}$$

In order to provide an accurate approximation of the target RT, and therefore to closely follow the A_{dB}, the attenuation filter used in the FDN implementation in the present study is a graphic equalizer (GEQ), which controls the energy decay of the system in ten octave bands, with center frequencies from 31.25 Hz to 16 kHz. The equalizer is composed of biquad filters [16] and designed with the method proposed by Välimäki and Liski [17] with later modifications, such as the scaling by a median of gains and the adding of a first-order high-shelf filter as proposed in [18]. The GEQ magnitude response for the ith delay line is expressed in dB as

$$\widetilde{H}_{\mathrm{dB},i}(e^{j\omega}) = g_0 + \sum_{m=1}^{M} \left(H_{\mathrm{dB},i,m}(e^{j\omega}) - \frac{g_0}{M} \right), \tag{4}$$

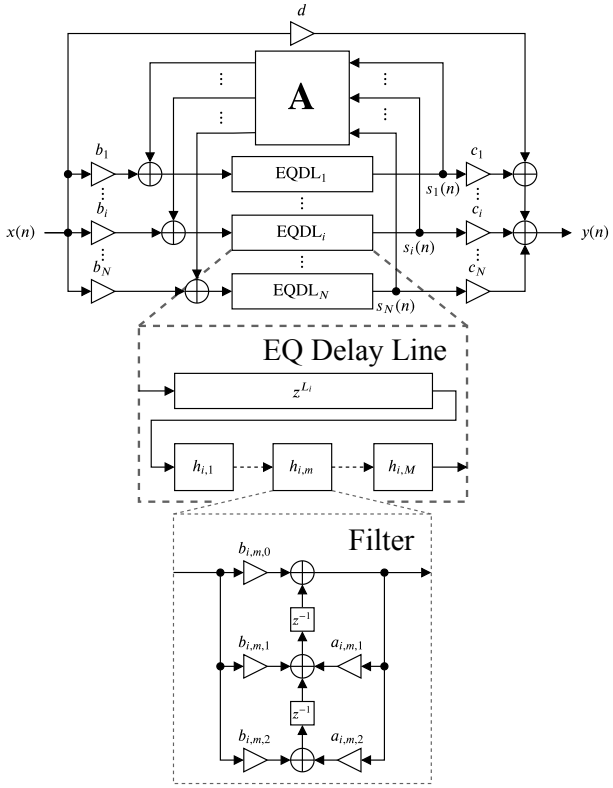


Figure 1: *Flow diagram of an FDN with N equalized delay lines and their M octave-band biquad filters shown in detail. See Sections 2 and 3.5 for more details.*

where g_0 is the broadband gain factor, $H_{\mathrm{dB},i,m}$ are the magnitude responses of the band filters, and $m = 1, 2, ..., M$ is the frequency-band index with M controlled frequency bands. The time-domain representation $\widetilde{h}_i(n)$ of $\widetilde{H}_{\mathrm{dB},i}(e^{j\omega})$ is used in Eq. (1b).

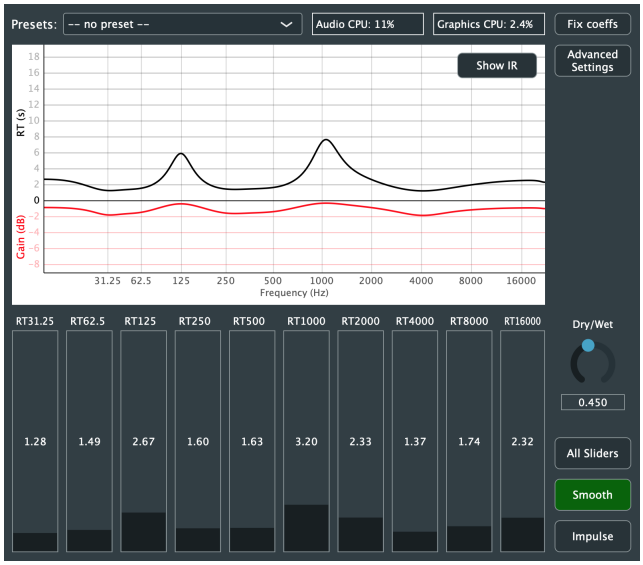
3. IMPLEMENTATION

This section describes the real-time implementation of late reverberation synthesis using an FDN and a modified GEQ as the attenuation filter. The algorithm has been implemented in the form of an audio plugin in C++ using JUCE, an open-source cross-platform application framework [19]. The plugin can be downloaded from [20], and an explanatory demo video can be found in [21].

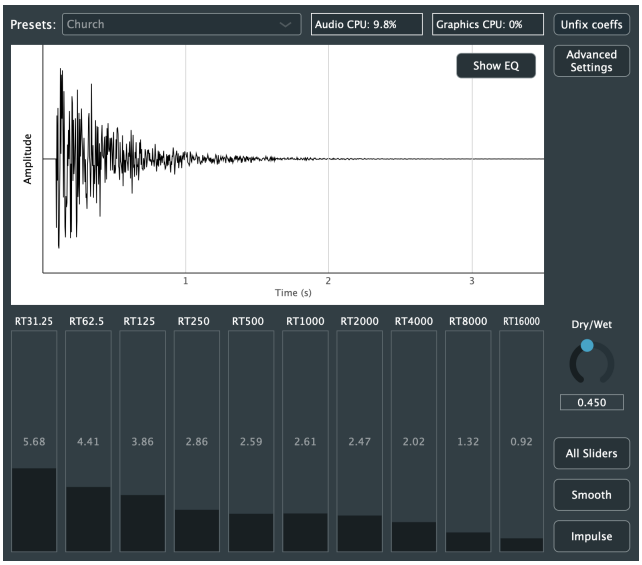
3.1. Control over RT Values

In the present implementation, the modified GEQ attenuation filter allows controlling the RT values in ten frequency bands. In order to utilize the whole potential of the filter, the GUI of the plugin is equipped with ten vertical sliders, one for each frequency band, as depicted in Fig. 2. By changing the value of each of the sliders, the user is able to change the RT value for the corresponding frequency band from 0.03 s to 15 s with a 0.01-s step.

Since too large a difference between two consecutive RT values can cause instability [18, 22], two extra modes are implemented for better control: the *All Sliders* and the *Smooth* modes.



(a) *The attenuation filter's response (red line) and the corresponding RT curve (black line). No preset is selected, and the* Smooth *button is pressed.*



(b) *Reverberator IR. The* Fix coeffs *button has been pressed, and the preset's drop-down menu and sliders have been disabled (see Sec. 3.6).*

Figure 2: *GUI of the implemented FDN plugin.*

The modes are activated by pressing the respective buttons, as indicated in Fig. 2a, with the corresponding buttons on the GUI being highlighted in green. If a mode is activated when the other is active, the latter will deactivate. The *All Sliders* mode allows the user to set all the RT values to be the same by changing the slider position in one of the frequency bands.

When the *Smooth* mode is activated, changing the value of one RT will also adjust the RT in the other frequency bands. RT values of bands closer to the band that is changed are more affected than other RT values via the formula

$$T_{60}[m] = T_{60,\text{init}}[m] + \Big(T_{60}[m_\text{c}] - T_{60,\text{init}}[m_\text{c}]\Big)\epsilon^{|m-m_\text{c}|}, \quad (5)$$

where m_c is the index of the currently adjusted slider, $m = 1, 2, ..., M$ is the slider number, T_{60} and $T_{60,\text{init}}$ are the final and initial RT values, respectively, and $\epsilon = 0.6$ is a heuristically chosen scaling factor.

Five typical reverberation presets were created: *Small Room*, *Medium Room*, *Large Room*, *Concert Hall*, and *Church*. The first three presets are based on the measurement results presented in [23], whereas RT values for the last two are taken from [24]. All examples are available in a drop-down list in the top part of the GUI. If one of the sliders is changed, "– no preset –" is displayed in the drop-down list, as shown in Fig. 2a.

The *Impulse* button at the bottom right of the GUI empties the delay lines and feeds a Dirac delta into the system so that the impulse response of the reverberator is produced as an output.

3.2. Response Plotting

The window in the upper-half part of the GUI displays plots that inform the user about the state of the plugin. As seen in Fig. 2a, the GUI can display the RT curve (black) and the corresponding magnitude response of the attenuation filter (red), which are plotted in real time based on the values set by the sliders. This provides the user with an insight into the actual decay characteristics of the synthesized reverberation, which may differ from the user-defined RT values. This happens due to the limited ability of the attenuation filter in following the target RT curve, especially when the differences between values set for the neighboring frequency bands are big [18, 22]. Very extreme differences may lead to the filter's magnitude response reaching or exceeding 0 dB, which results in the system's instability. This state is signaled by the background color of the window changing to light red. For the response, only one delay line is used to retain real-time plotting. Due to the fact that the attenuation filter adopts smaller values for shorter delay-line lengths, the shortest delay line is chosen as it exhibits instability sooner than the others.

The *Show IR* button located in the top right of the window allows the user to toggle between the RT curve and filter's response plots and the reverberator's IR plot, which is shown in Fig. 2b. As opposed to the response, the longest delay line is used to calculate the IR. Even though the effect of the scattering matrix, and with that the effect of other delay lines, are not included, using the longest delay line has been proven empirically to give a good indication of the audible IR. The values displayed on the x-axis are determined by the average slider value, i.e., a shorter reverb time, results in a more detailed plot of the earlier seconds of the IR. Furthermore, not every sample is drawn, but 1,000 data points spread over the plot-range.

3.3. Choice of Delay Lengths and Distribution

Although FDN-based reverbs are nowadays among the most popular algorithmic reverbs, there is no clear rule on how to choose the lengths of the delay lines [25, 26]. The common practice is to choose the number of samples that are mutually prime and uniformly distributed between the maximum and minimum lengths to avoid clustering of echoes [26].

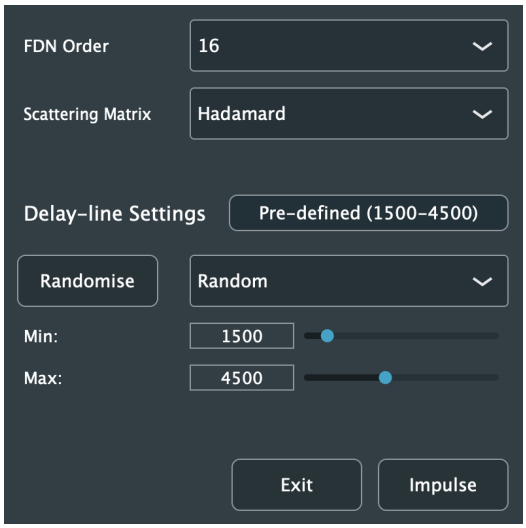


Figure 3: *The advanced settings window.*

Through the *Advanced Settings* window shown in Fig. 3, the distribution of delay-line lengths can be chosen through a drop-down menu from four options: *Random*, *Gaussian*, *Primes*, and *Uniform*. Whenever an option is selected, the delay-line lengths are randomly generated based on the distribution selected and rounded to the nearest integers. The generation can be repeated by clicking on the *Randomize* button. Furthermore, the minimum (500 samples) and maximum (10,000 samples) delay-line lengths can be controlled; the minimum difference between the two has been set to 100 samples. Moreover, there is an option to have the lengths pre-defined for each distribution so that the plugin will have the same behavior every time it is used. The minimum and maximum delay-line lengths have been empirically set to 1,500 and 4,500 samples, respectively (~30–100 ms at $f_\mathrm{s} = 44.1$ kHz).

3.4. Choice of Feedback Matrix

The choice of the feedback matrix is crucial for the FDN algorithm to work correctly. The popular matrix types used in FDN implementations that fulfill the requirement of being unilossless are Hadamard [27], Householder [27], random orthogonal, and identity matrices [28]. Where the first three are chosen to enhance specific properties of the algorithm, e.g., density of the impulse response, the identity matrix, however, reduces the FDN to a Schroeder reverberator, or a parallel set of comb filters [6, 28]. The plugin presented in this study allows the user to choose between these four matrices through a drop-down menu and to learn about the differences in the sound obtained by changing this part of the FDN reverberator. Additionally, the order of the FDN, and thus the size of the feedback matrix, can be changed. The available options are 2, 4, 8, 16, 32, and 64, which can be chosen from a menu.

In the case of the Householder matrix type, the implementation of matrices of different sizes vary. For all orders except for 16, the matrix is constructed using following the formula:

$$\mathbf{A}_N = \mathbf{I}_N - \frac{2}{N}\mathbf{u}_N\mathbf{u}_N^\mathrm{T}, \tag{6}$$

where $\mathbf{u}_N^\mathrm{T} = [1, \ldots, 1]$, and $\mathbf{I}_N$ is the identity matrix [27]. The matrix of order 16, on the other hand, following [29], is constructed using the recursive embedding of matrix of order 4:

$$\mathbf{A}_{16} = \frac{1}{2}\begin{bmatrix} \mathbf{A}_4 & -\mathbf{A}_4 & -\mathbf{A}_4 & -\mathbf{A}_4 \\ -\mathbf{A}_4 & \mathbf{A}_4 & -\mathbf{A}_4 & -\mathbf{A}_4 \\ -\mathbf{A}_4 & -\mathbf{A}_4 & \mathbf{A}_4 & -\mathbf{A}_4 \\ -\mathbf{A}_4 & -\mathbf{A}_4 & -\mathbf{A}_4 & \mathbf{A}_4 \end{bmatrix}. \tag{7}$$

As a result, the matrix of order 16 consists of the same values, differing only in their sign.

3.5. Code Structure

The plugin is divided into two main components that run on different threads at different rates. Firstly, the DSP component running at 44,100 Hz (audio rate), is structured in the same fashion as shown in Fig. 1. An *FDN* class contains the scattering matrix $\mathbf{A}$, vectors $\mathbf{b}$ and $\mathbf{c}$ that scale the inputs and outputs of each delay line (marked as b_i in Eq. (1b) and c_i in Eq. (1a), respectively, and in the current implementation all set to 1), and N instances of the *EQDelayLine* class. This class, in turn, contains a delay line of length L_i (implemented as a circular buffer) and M instances of the *Filter* class. This class does all the low level computation and contains the filter states and coefficients $b_{i,m}$ and $a_{i,m}$ of the ith delay line and the mth octave band.

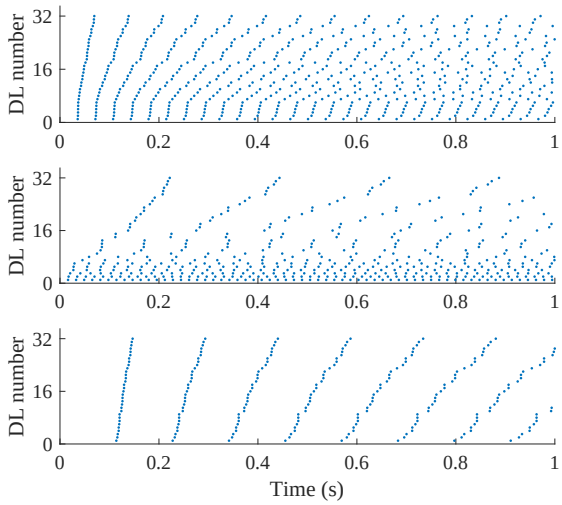
Secondly, the GUI component running at 5 Hz is responsible for the graphics and control of the FDN. Apart from the controls, this component contains the *Response* class that is used to draw the RT and gain curves and the IR shown in Figs. 2a and 2b. The filter coefficients necessary for drawing the curves are updated at the aforementioned rate. This calculation also provides information about the stability of the FDN and is used to trigger the light-red background denoting instability. The *Response* class also contains a single instance of the *EQDelayLines* class that is used to calculate the IR.

Communication from the GUI to the DSP component happens at a 5-Hz control rate, which has been found to be a great trade-off between speed and quality of control. When changing any of the non-RT controls, the GUI triggers flags that are outside of the process buffer (512 samples) to avoid the manipulation of parameters when sample-by-sample calculations are being made.

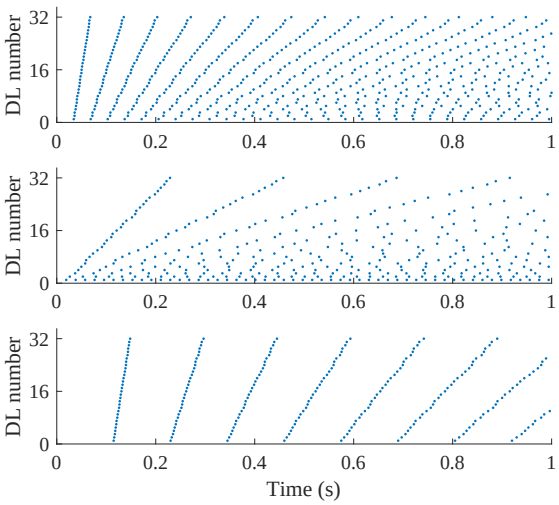
3.6. Real-time Considerations

The components of the plugin requiring most computations are the (re-)calculation of the filter coefficients and the plotting of the responses. Even though the filter coefficients only need to be recalculated when the sliders' values are changed, it is good practice for a plugin to have the same CPU usage when its values are changed as when its values are static to prevent unexpected spikes in the CPU usage. Instead, a *Fix coeffs* (coefficients) button has been implemented that, when clicked, will deactivate the preset's drop-down menu and the sliders (as shown in Fig. 2b). Furthermore, the plugin will stop recalculating the plots and filter coefficients, greatly decreasing CPU usage (see Sec. 4.2). The CPU usages of both threads are shown at the top of the plugin.

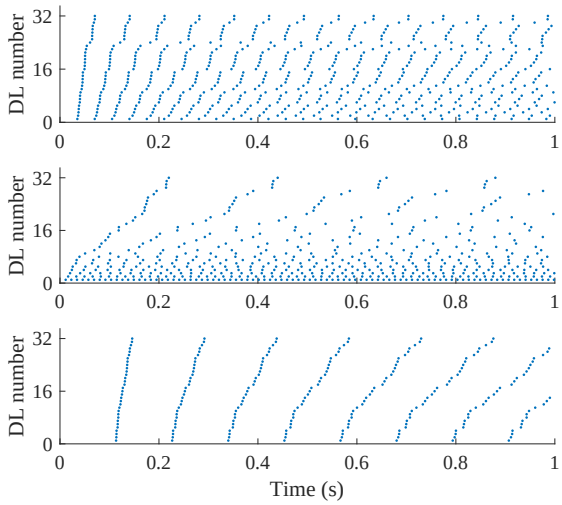
When any change is made to the FDN, be it the order, delay-line distribution or length, the delay lines and filter states are set to zero to prevent any unwanted artifacts. Only the RT control works in real time without emptying the delay lines and filter states.



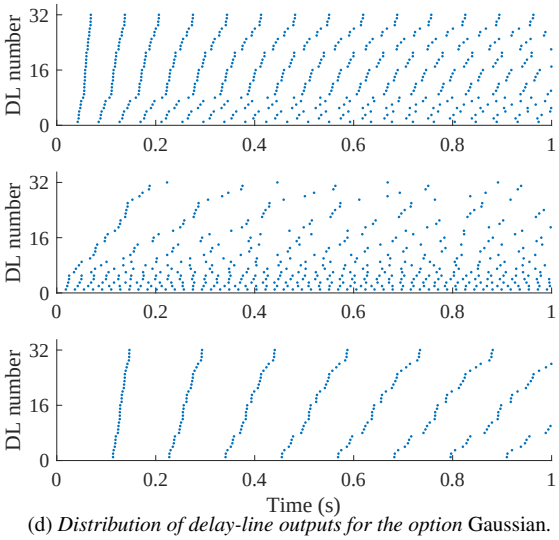
(a) *Distribution of delay-line outputs for the option* Primes.



(b) *Distribution of delay-line outputs for the option* Uniform.



(c) *Distribution of delay-line outputs for the option* Random.



(d) *Distribution of delay-line outputs for the option* Gaussian.

Figure 4: *Distribution of the outputs of 32 delay lines (without scattering) for the options (a) primes, (b) random, (c) uniform, and (d) Gaussian, and the length range (top) pre-defined lengths (1,500–4,500 samples), (middle) lengths randomized over the entire range (500–10,000 samples), and (bottom) lengths randomized over a narrow range (5,000–6,500 samples). Each dot marks the time when the given delay line outputs a sample.*

4. RESULTS AND DISCUSSION

This section presents results regarding the echo density produced by and CPU usage of the plugin.

4.1. Echo Density

To achieve smooth reverberation, a sufficient echo density, i.e., the number of echoes per time unit produced by the algorithm and their distribution [26], should be obtained. Echo density is affected by a few factors, such as the lengths and the distribution of the delay lines, the type of the feedback matrix [30] and its size, all of which are discussed below.

4.1.1. Delay Lengths

The choice of delay-line length-distribution can help avoid more than one sample appearing at the system's output at the same time and a clustering of the echoes, since both of these phenomena lower the echo density. Additionally, the range over which the delay-line lengths are chosen also affects the quality of the synthesized sound. The distribution of delay-line outputs over time, without a scattering matrix (i.e., an identity feedback matrix is used), is shown in Fig. 4 for all the options available in the plugin. In the case of the randomized selection of the delay-line lengths (middle and bottom panes of Figs. 4a–4d), the results show one of the possible configurations. The delay-line lengths used in the examples

were sorted in ascending order.

The top panes of Figures 4a–4d show the outputs of the predefined delay lines, which depict the typical behavior of the FDN algorithm. The outputs become more diffused over time, making the reverb smoother. It should be noticed, however, that when using *Uniform* distribution, the chosen range is divided into portions proportional to the FDN order, and the delay-line lengths are chosen from such "bands". This makes the consecutive delay lines differ by a similar number of samples, and the possibility of output samples overlapping or clustering is higher than with other distributions. Choosing the *Gaussian* option, on the other hand, draws the delay-line lengths from the normal distribution with the mean being the midpoint between the range's boundaries. This results in choosing the lengths closer to the mean more often than those further from it, as depicted in Fig. 4d, potentially causing clustering of echoes and slowing down the increase of the echo density.

The distribution of outputs presented in the middle panes show that when the delay-length range is very wide, the output is diffused from the beginning. Since such decay is rarely met in reality, it is useful when recreating only specific spaces [31]. Additionally, very short delay lines create clusters of echoes and a huge portion of the output samples overlap. They do not contribute to the increase of echo density, but nevertheless add to the computation. Such clusters are well visible in Figs. 4a and 4c. Moreover, the attenuation applied to the short delay lines is usually small, and therefore closer to 0 dB, which makes them more prone to causing the system's instability.

On the other hand, very long delay lines (10,000 samples translates to about 0.23 s for the 44.1-kHz sample rate) may not produce a meaningful contribution to the synthesized reverb for low RT values. However, such long delay lines still add to the computation, since the order of the FDN, and at the same time, the size of the feedback matrix needs to be equal to the number of delay lines.

Using a very narrow range over which the delay-line lengths are distributed results in clusters of samples arriving at the output within a very short time, as seen in the bottom panes of Figs. 4a–4d. Between the consecutive clusters, however, relatively long silences occur. The synthesized reverberation tail diffuses very slowly. Regardless of whether the delay-line lengths are chosen to be prime, random, distributed normally or uniformly, choosing too narrow a range results in low sound quality with clearly audible segmentation and in the effect's behavior resembling more that of a single delay line than a reverb.

4.1.2. Feedback Matrix

The normalized echo densities for all types of matrices available in the plugin were calculated, following the method presented in [32, 33, 34], for orders 2–64 and the delay lines selected randomly from the range between 1,500 and 4,500 samples (the same set of delay lines was used for all calculations). To avoid bias caused by the smearing of echoes due to the filtering, the attenuation filters were not used in the calculations. The results are presented in Fig. 5 which generally show that the echo density increases faster with a higher FDN order than with a lower one.

When matrices of size 2 and 4 are used, the number of echoes in the output of the reverberator increases slowly and may never reach saturation, i.e., the moment when there is an echo at every successive time unit [26]. Therefore, these low orders do not produce smooth sound. In the case of an FDN of order 8, the echo density build-up is slow, which results in audible artifacts in synthesized reverbs for as long as one second. Thus, a matrix of size 16 is the smallest that increases the number of echoes quickly enough so that the resulting sound is perceived as smooth for all matrix types (except for the identity matrix). For the Hadamard and random matrices, a further rise in the size accelerates the echo density build-up, as evident in Fig. 5c and 5d.

Interestingly, the Householder matrix excels with the order of 16 using the recursive embedding of Eq. (7). This can be explained by the fact that for all other orders, the implementation follows Eq. (6), which produces matrices in which the difference between the diagonal and the rest of the elements grows proportionally to the order. Effectively, this makes the FDN approach a bank of decoupled comb filters, which results in high variability of echo density for orders 32 and 64, as seen in Fig. 5b, leading to audible artifacts in the reverberation. For the matrix of order 16, however, the echo density increases fast and remains high once saturation is reached.

Because the identity matrices produce a very low echo density that does not increase with time, as seen is Fig. 5a, they are not well fitted for the FDN. Reverberation synthesized using such matrices is always low-quality. Being also an identity matrix, the Householder matrix of order 2 should be avoided as well.

Table 1: *CPU usage for all FDN orders in the cases of unfixed (plotting IR and EQ) and fixed coefficients.*

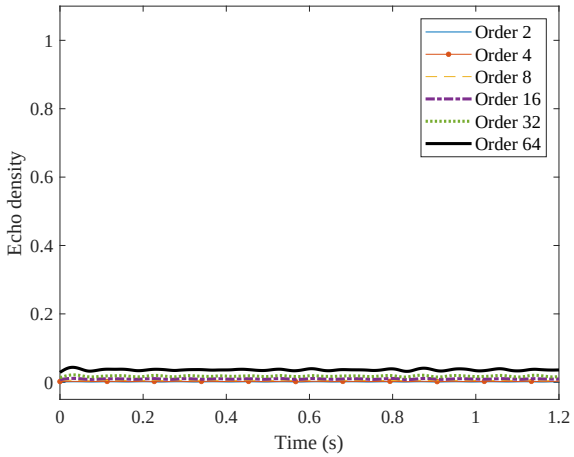
FDN Order	CPU usage (%)		
	Unfixed (IR)	Unfixed (EQ)	Fixed
2	18.4	11.0	**3.1**
4	19.8	12.0	**5.4**
8	22.7	15.2	**7.9**
16	28.6	22.2	**13.3**
32	46.1	40.2	**30.4**
64	110.5	100.1	**92.5**

4.2. CPU Usage

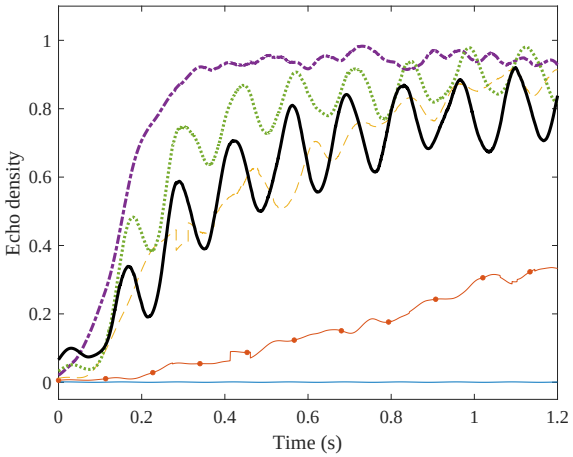
Table 1 shows the CPU usage for all implemented FDN orders for three different plugin-states: unfixed coefficients plotting the IR, unfixed coefficients plotting the EQ, and fixed coefficients (plotting and recalculation of filter coefficients disabled). The performance has been measured on a MacBook Pro with a 2.2 GHz Intel i7 processor using *Xcode*'s time profiler [35].

For all plugin states, the CPU usage increases exponentially with the FDN order. Furthermore, fixing the coefficients, and thus disabling the plotting and filter-coefficient calculation, greatly decreases the plugin's CPU usage. Comparing this to the unfixed EQ case, an additional ~8.0% is added to the CPU usage, and when plotting the IR versus the EQ, an additional ~7.5% is added to the usage. This value, however, depends on the average reverb time used. For testing, the *Concert hall* preset was used, which requires calculating 2.5 s of sound for the IR plot. With a higher average slider value, and thus a longer IR to be calculated and plotted, the CPU usage also increases.

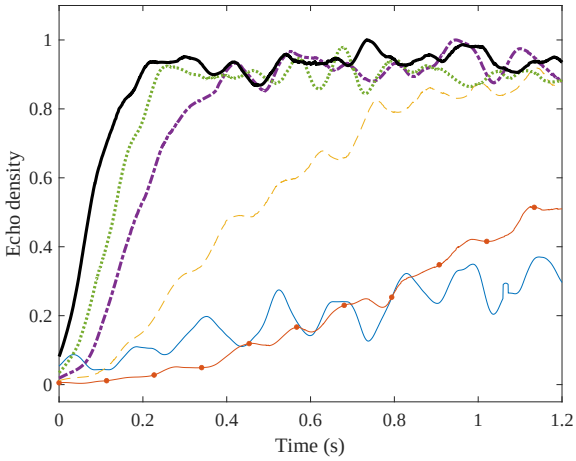
The smallest useful FDN order is 16, as stated in Sec. 4.1.2. Table 1 shows that this order, or even 32, is unlikely to cause auditory drop-outs, especially when the coefficients are fixed.



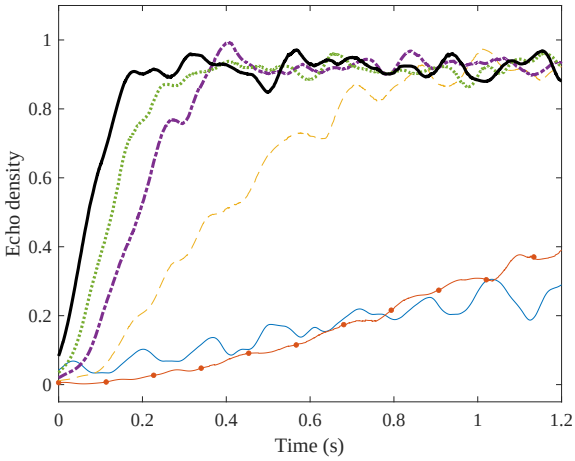
(a) *Identity matrix.*



(b) *Householder matrix.*



(c) *Hadamard matrix.*



(d) *Random orthogonal matrix.*

Figure 5: *Normalized echo densities for four types of feedback matrices and different FDN orders.*

5. CONCLUSIONS

The present study introduces the FDN-based artificial reverberation synthesis plugin. The implementation allows control over the decay characteristics of the sound in ten octave bands in real timeand plots the corresponding RT curve, the attenuation filter's magnitude response, and the IR. Additionally, users can explore different setups of the FDN by changing the type and size of the feedback matrix, and the lengths and distribution of the delay lines.

Experiments with the delay-line lengths and their distributions suggest that these parameters should always be used in a balanced manner, that suit the target reverberation. A wrong choice may result in the creation of clusters of output samples and a low echo density, which is undesirable in a reverberator. Choosing the lengths over a narrow range results in low-quality, segmented sound, which diffuses slowly. Picking the right distribution of delay-line lengths is also important.

The ability to choose from among different FDN orders shows that the lowest useful order for high-quality sound processing is 16, as it a sufficiently provides fast echo density build-up to obtain smooth reverberation without audible artifacts. Shifting between feedback matrix types proves that the identity matrix, even though it is lossless, should not be used in such applications, since the produced sound is fluttery. It also shows that, in the case of the Householder matrix, implementation affects the reverberation. Results show that using recursive embedding when constructing the Householder matrix increases the echo density in the produced reverberation.

6. ACKNOWLEDGMENTS

This work was initialized, when Karolina Prawda made a Short-Term Scientific Mission to the Aalborg University Copenhagen from October 28 to November 15, 2019.

7. REFERENCES

[1] M. R. Schroeder and B. F. Logan, "Colorless artificial reverberation," *J. Audio Eng. Soc.*, vol. 9, no. 3, pp. 192–197, Jul.

1961.

[2] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421–1448, Jul. 2012.

[3] N. Peters, J. Choi, and H. Lei, "Matching artificial reverb settings to unknown room recordings: A recommendation system for reverb plugins," in *Proc. Audio Eng. Soc. 133rd Conv.*, San Francisco, CA, USA, Oct. 2012.

[4] C. Kereliuk, W. Herman, R. Wedelich, and D. J. Gillespie, "Modal analysis of room impulse responses using subband ESPRIT," in *Proc. 21st Int. Conf. Digital Audio Effects*, Aveiro, Portugal, 4–8 Sept. 2018.

[5] S. Bilbao and B. Hamilton, "Passive volumetric time domain simulation for room acoustics applications," *J. Acoust. Soc. Am.*, vol. 145, no. 4, pp. 2613–2624, Apr. 2019.

[6] J. M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. 90th Audio Eng. Soc. Conv.*, Paris, France, 19–22 Febr. 1991.

[7] J. M. Jot and A. Chaigne, "Method and system for artificial spatialisation of digital audio signals," Feb. 1996, U.S. Patent 5,491,754.

[8] S. Heise, M. Hlatky, and J. Loviscach, "Automatic adjustment of off-the-shelf reverberation effects," in *Proc. Audio Eng. Soc. 126th Conv.*, Munich, Germany, 7–10 May 2009.

[9] C. Borß, "A VST reverberation effect plugin based on synthetic Room Impulse Responses," in *Proc. 12th Int. Conf. on Digital Audio Effects (DAFx-09)*, Como, Italy, 1–4 Sept. 2009.

[10] Ableton, "Convolution reverb," Available online at http://www.ableton.com/en/packs/convolution-reverb/, Accessed: 2020-03-16.

[11] S. Philbert, "Developing a reverb plugin; utilizing Faust meets JUCE framework," in *Proc. Audio Eng. Soc. 143rd Conv.*, New York, NY, USA, 18–21 Oct. 2017.

[12] D. Moffat and M. B. Sandler, "An automated approach to the application of reverberation," in *Proc. Audio Eng. Soc. 147th Conv.*, New York, NY, USA, 16–19 May 2019.

[13] T. Erbe, "Building the Erbe-Verb: Extending the feedback delay network reverb for modular synthesizer use," in *Proc. Int. Computer Music Conf.*, Denton, TX, USA, Sept. 2015.

[14] Valhalla DSP, "Valhalla Room," Available online at http://valhalladsp.com/shop/reverb/valhalla-room/, Accessed: 2020-03-31.

[15] S. J. Schlecht and A. P. Habets, "On lossless feedback delay networks," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1554–1564, Mar. 2017.

[16] S. J. Orfanidis, *Introduction to Signal Processing*, Rutgers Univ., Piscataway, NJ, USA, 2010.

[17] V. Välimäki and J. Liski, "Accurate cascade graphic equalizer," *IEEE Signal Process. Lett.*, vol. 24, no. 2, pp. 176–180, Feb. 2017.

[18] K. Prawda, S. J. Schlecht, and V. Välimäki, "Improved reverberation time control for feedback delay networks," in *Proc. 22nd Int. Conf. Digital Audio Effects*, Birmingham, UK, Sept. 2019.

[19] ROLI, "JUCE," Available at http://juce.com/, Accessed: 2020-04-03.

[20] S. Willemsen, "FDN plugin github release v1.0," Available at https://github.com/SilvinWillemsen/FDN_/releases/tag/v1.0, Accessed: 2020-03-19.

[21] S. Willemsen, "Real-time FDN," Available online at https://youtu.be/ddgKMtW1Obc, Accessed: 2020-03-19.

[22] S. J. Schlecht and A. P. Habets, "Accurate reverberation time control in Feedback Delay Networks," in *Proc. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, 5–9 Sept. 2017, pp. 337–344.

[23] M. Jeub, M. Schäfer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proc. Int. Conf. Digital Signal Process. (DSP)*, Santorini, Greece, Jul. 2009, pp. 1–4.

[24] Audiolab University of York, "Open AIR library," Available at http://openairlib.net/, Accessed: 2020-04-07.

[25] D. Rocchesso and J. O. Smith, "Circulant and elliptic feedback delay networks for artificial reverberation," *IEEE Trans. Speech and Audio Process.*, vol. 5, no. 1, pp. 51–63, Jan. 1997.

[26] S. J. Schlecht and E. A. P. Habets, "Feedback delay networks: Echo density and mixing time," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 25, no. 2, pp. 374–383, Feb. 2017.

[27] J. M. Jot, "Efficient models for reverberation and distance rendering in computer music and virtual audio reality," in *Proc. Int. Computer Music Conf.*, Thessaloniki, Greece, Sept. 1997.

[28] F. Menzer and C. Faller, "Unitary matrix design for diffuse Jot reverberators," in *Proc. Audio Eng. Soc. 128th Conv.*, London, UK, May 22–25 2010.

[29] J. O. Smith, *Physical Audio Signal Processing*, `http://-ccrma.stanford.edu/~jos/pasp/`, Accessed 2020-04-17, online book, 2010 edition.

[30] O. Das, E. K. Canfield-Dafilou, and J. S. Abel, "On the behavior of delay network reverberator modes," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoustics (WASPAA)*, New Paltz, NY, USA, Oct. 2019, pp. 50–54.

[31] S. Oksanen, J. Parker, A. Politis, and V. Välimäki, "A directional diffuse reverberation model for excavated tunnels in rock," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Vancouver, Canada, May 2013, pp. 644–648.

[32] J. S. Abel and P. Huang, "A simple, robust measure of reverberation echo density," in *Proc. Audio Eng. Soc. 121st Conv.*, San Francisco, CA, USA, Oct. 2006.

[33] P. Huang and J. S. Abel, "Aspects of reverberation echo density," in *Proc. Audio Eng. Soc. 123rd Conv.*, New York, NY, USA, Oct. 2007.

[34] P. Huang, J. S. Abel, H. Terasawa, and J. Berger, "Reverberation echo density psychoacoustics," in *Proc. Audio Eng. Soc. 125th Conv.*, San Francisco, CA, USA, Oct. 2009.

[35] Apple Inc., "Xcode – Apple Developer," Available at https://developer.apple.com/xcode/, Accessed: 2020-03-18.